An Efficient Method for Quick Construction of Web Services
Hao Han, Yohei Kotake and Takehiro Tokuda
{han, kotake, tokuda}@tt.cs.titech.ac.jp
Department of Computer Science, Tokyo Institute of Technology
Meguro, Tokyo 152-8552, Japan

Abstract. With the development of the Internet, Web services, such as Google Maps API and YouTube Data API, become more important and convenient for the Web knowledge distribution and integration. However, currently, most existing Web sites do not provide Web services. In this paper, we present a partial information extraction technology to construct the Web services from the general Web applications quickly and easily. Our implementation shows that our approach is applicable to different kinds of Web applications.

Keywords: Web application, Web service, information extraction

1 Introduction

Web services are XML-based software systems designed to support interoperable machine-to-machine interaction over networks, and executed on remote systems hosting the requested services. By using Web services, Web sites can publish their functions or messages to the rest of the world. Many Web sites, such as Google, YouTube and Flickr, provide their Web services like online maps, videos and pictures, and these Web services are widely used for Web information integration and popular with the mashup system development.

However, unfortunately, most existing Web sites do not provide Web services. For the Web sites, Web applications are still the main methods for the information distribution by the Web documents in a standard format supported by common browsers such as HTML and XHTML. For example, CNN [2] lets users search for the online news by inputting the keywords at Web page. Once the users submit the search, CNN would present the news search results. However, this news search service can not be integrated because CNN does not open this application by a Web service. Similarly, Wikipedia does not provide the official Web service APIs and it is difficult for the developers to integrate it with other Web services.

In this paper, we propose a partial information extraction approach to construct the Web services. We design the extraction patterns for the target Web sites, and use these patterns to extract the partial information from Web documents to create the resulting tables, finally respond to the requests of users with corresponding field values like the real Web services. We run the actual extraction and query processes of the constructed Web services at a proxy server, and provide the designed Web service interfaces.

The organization of the rest of this paper is as follows. In Section 2 we give an overview of our Web service construction approach. In Section 3 we explain the method of partial information extraction, information query and interface configuration in detail. We give the
implementation and evaluation of our approach in section 4. In Section 5 we give related work, and finally conclude in Section 6.

2 Overview

A real Web service responds to the requests from users by returning the data from server-side information resources, usually a database. For our Web service construction approach, the information resources are the Web applications. We need to extract the information from the Web applications to generate the resulting tables like the tables of database.

We give an overview of our Web service construction approach. Our approach is based on the partial-information-extraction-method and resulting-table-query-method as described in the following steps:

Firstly, we select the target parts from Web pages to generate the extraction patterns, which comprise the names and data types for the selected parts.

Secondly, we run the constructed Web services at a proxy server, and configure Web service interfaces for them.

Thirdly, the Web services use the extraction patterns to extract the partial information statically or dynamically according to the requests of users to create the resulting tables.

Finally, the Web services search for the desired information from the resulting tables and respond to the users with the corresponding field values.

We explain our approach in detail in the following sections.

3 Web Information Extraction Method

Not all the contents of Web applications are necessary and useful for Web service construction. As the Web service developers, we need to select the available information from the Web pages according to the actual function of the constructed Web service. For example, we
just select the result list from a search result page except the parts such as advertisements if we want to construct a Web service to realize the search function.

We design the extraction pattern for each target Web application and use the extraction pattern to acquire our selected partial information statically or dynamically. Each HTML document of Web application can be parsed as a tree structure, and our extraction method is based on the analysis of the tree structure.

3.1 Extraction Pattern

Our extraction pattern comprises two parts: data type and path. Data type represents "What kind of information is needed?" and path represents "Which part of document is needed?". We define a field name and data type for each target part, then acquire the path of the part.

3.1.1 Data Types Definition

There are many kinds of information in Web applications such as a piece of text or a group of photos. In order to extract information accurately and select the target parts conveniently, we define a name for each target part and its data type for the extraction of partial information. The data type includes two kinds of information: property and structure.

Property is text, object or link. Text is the character string in Web pages such as an article. Object is one instance of the photo, video and other multimedia file. Link is a reference in a hypertext document to another document or other resource.

Structure is single occurrence or continuous occurrence. A single occurrence is a node without similar sibling nodes such as the title of an article, and the continuous occurrence is a list of nodes with similar paths such as result list in a search result page.

There are six kinds of data types: single text, continuous text, single object, continuous object, single link and continuous link. For example, for a news article Web page shown in Fig. 2, the news title is a single text with name Title, the news contents are continuous text with name Paragraph, one photo is a single object with name Photo and the related news are shown as continuous link with name Related Link.

3.1.2 Parts Selection

We select the target parts to reach the partial information. Each target part is represented by a node, and each node can be represented by its path from the root. We use the following form to save the path of a selected part:

\[
\text{body} : 0 : \text{ID}/N_1 : O_1 : \text{ID}_1/N_2 : O_2 : \text{ID}_2/\ldots/N_{n-1} : O_{n-1} : \text{ID}_{n-1}/N_n : O_n : \text{ID}_n
\]

Where, \(N_n\) is the node name of the \(n\)-th node, \(O_n\) is the order of the \(n\)-th node among the sibling nodes, \(\text{ID}_n\) is the ID value of the \(n\)-th node, and \(N_{n-1}\) is the parent node of \(N_n\).

For a group of parts whose structures are continuous occurrence, we do not need to select all the parts one by one and just select one of them. For example, in a search result page containing ten result items, we just select the first item and define the structure as continuous occurrence for it.
3.2 Partial Information Extraction

We use the extraction pattern that contains defined data types and acquired paths to realize the partial information extraction.

3.2.1 Similar Paths

For a Web site, the response Web pages are similar to each other in common if the requests are similar and sent to the same target because there are a number of Web pages dynamically generated by the same server-side programs. For example, in the BBC Country Profiles site [12], there exists a collection of 200 or more country/region information including most recent basic information such as capital city, population and area information. If we choose a country, the Web page of this country would be returned, and all the country profile pages are similar to each other as shown in Fig. 3. So, the paths of the similar parts of these country profile pages are similar to each other, too.

**Similar Path of Part:** Two paths are similar to each other, if these two paths have the same forms ignoring the difference of orders of nodes among sibling nodes, and the difference of orders is within a defined deviation range. The form of path is as follows:

\[
\text{body} : 0 : ID_1/N_1 : (O_1 - h \sim O_1 + h) : ID_2/N_2 : (O_2 - h \sim O_2 + h) : ID_3/.../N_{n-1} : (O_{n-1} - h \sim O_{n-1} + h) : ID_n/N_n : (O_n - h \sim O_n + h) : ID_n
\]

Where, \( N_n \) is the node name of the \( n \)-th node, \( O_n \) is the order of the \( n \)-th node among the sibling nodes, \( ID_n \) is the ID value of the \( n \)-th node, \( N_{n-1} \) is the parent node of \( N_n \), and \( h \) is the deviation value. An example of similar paths is shown in Fig. 4.

We use the ID value to choose the most appropriate paths with the minimum deviation value from the deviation range.

Although the layout of Web pages is relatively steady and not changed randomly usually, some Web sites change the layout of Web pages periodically or irregularly. The similar paths could find the correct target parts automatically after the Web sites change the layout of Web pages if the changed paths of target parts are in the deviation ranges.
3.2.2 Node List Extraction

In the tree structure of HTML document, each path represents a node. We extract the nodes according to the corresponding paths. If the data type of a part is continuous occurrence, we use the following steps to extract the list of nodes:

1. We use the path to find the corresponding node $N$.
2. We get the parent node $P$ of $N$.
3. We get the subtree $S$ whose root node is $P$.
4. We get the node list $L$ of which each has the same path as $N$ without considering the orders of child nodes of $P$ under $S$.
5. If we find two or more than two nodes in $L$ and these nodes are different from the nodes of other selected parts, or $P$ is $<body>$, then $L$ is the final node list. Otherwise, we set the parent node of $P$ as the root node of $S$, then go to Step 4.
Each node of the extracted node list $L$ represents a part of continuous parts as shown in Fig. 5.

The table is a kind of often-used information for Web service construction. There are two types of tables in Web pages: horizontal type and vertical type. Each column of a horizontal type table is usually identified by a column name and the first row is the header row to display the column names. Each row of a vertical type table is usually identified by a row name and the first column is the header column to display the row names. If the users want to construct a Web service based on a table in a target Web page, they need to select the first cell of each column of a horizontal table or the first cell of each row of a vertical table, and set the structure as *continuous occurrence* for them. Fig. 6 shows the brief steps of node list extraction from tables.

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>Apple</td>
<td>100</td>
</tr>
<tr>
<td>002</td>
<td>Orange</td>
<td>150</td>
</tr>
<tr>
<td>003</td>
<td>Banana</td>
<td>120</td>
</tr>
</tbody>
</table>

Figure 6: Parts selection and nodes extraction from tables
3.2.3 Extraction Result

According to the defined data types, we extract the partial information from the extracted nodes in text format excluding the tags of HTML document as described in Table 1.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Partial Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>single text</td>
<td>node value of corresponding single leaf node</td>
</tr>
<tr>
<td>single object</td>
<td>attribute value of corresponding single node</td>
</tr>
<tr>
<td>single link</td>
<td>embedded link value of corresponding single node</td>
</tr>
<tr>
<td>continuous text</td>
<td>leaf node values of corresponding list of nodes</td>
</tr>
<tr>
<td>continuous object</td>
<td>attribute values of corresponding list of nodes</td>
</tr>
<tr>
<td>continuous link</td>
<td>link values of the list of nodes</td>
</tr>
</tbody>
</table>

For example, the extracted information of a photo is the value of attribute src of node `<img>`, and the extracted information of a link is the value of attribute href of node `<a>`.

Fig. 7 shows the extraction result of Yahoo Finance [3]. The extracted partial information is a document in XML format. The node name is the defined name of target part and the node value is the extracted information.

![Figure 7: Extraction result in XML format](image)

4 Web Service Construction and Configuration

We run our constructed Web services at a proxy server. In the proxy server, the extraction patterns are used to extract the partial information and the resulting tables are generated. By a designed interface, the users send the requests to proxy server and get the responses from the resulting tables.
4.1 Extraction Frequency and Mode

Usually, for the users of a real Web application that is suitable for Web service construction, there are two basic types of methods to send their requests. The first type is to enter a keyword into an input field by keyboard and click the submit button by mouse to send the query. The second type is to click a link or an option in drop-down list in Web page by mouse to view a new Web page. For the first type, the target Web pages are generated dynamically in server-side. The URLs of target Web pages are changed by the requests of users usually, and We call them Dynamic URLs. We use HtmlUnit \(^1\) to simulate the submit processes after the users send the requests and get the target Web pages. For the second type, the URLs are not changed by the requests of users. We call them Static URLs.

In the Web applications, some information is static or changed periodically. We do not need to extract them dynamically, especially the static information with static URLs, after the users send the requests every time. We define three kinds of extraction frequencies for Web services: dynamic extraction, periodic extraction and static extraction.

1. Dynamic Extraction: The Web services extract the partial information dynamically after the users send the requests. It is suitable for the information extraction from real time system or the search result Web pages with dynamic URLs such as CNN news search result pages.

2. Periodic Extraction: The Web services update the resulting table by extracting the partial information at a regular interval such as one hour, one day or one week. For example, the weekly ranking of hot items is updated once a week at Rakuten \([13]\).

3. Static Extraction: The extracted information is unchanged in a long period such as the basic information of a country/region. For example, the capital city and area information of most countries are unchanged and the population and life expectancy information remain unchangeable in a long period at BBC Country Profiles.

We can set the extraction frequencies for the constructed Web services according to the actual update frequencies of target Web applications.

4.2 Interface Configuration

Web services are self-contained and self-describing, and communicate using open protocols like HTTP. The most-used style architectures of Web services are SOAP and REST. SOAP stands for Simple Object Access Protocol. Google implements their Web services to use SOAP, and we can find SOAP Web services in a number of enterprise software. REST stands for Representational State Transfer. A number of new web services are implemented using a REST style architecture these days rather than a SOAP one, such as the Web services of Yahoo, Flickr and del.icio.us. Compared with SOAP, REST is lightweight and easy to build, and provides the readable results.

In the proxy server, we use the standard RESTful Web service interfaces between the users and the constructed Web services. The users can use the Web services by sending the proper parameter values to the Web services and get the response data through them.

\(^1\)http://htmlunit.sourceforge.net
For the interfaces between the constructed Web services and the target Web applications, if the target Web applications use dynamic URLs, we get the requests from the users and convert the requests to the appropriate forms, and send the converted requests to the target Web sites.

4.3 Resulting Table and Information Query

Usually, a real Web service can access the server-side database and search for the data in tables to create the response results. For our Web service construction approach, the extracted information comprises our virtual database. Like a table of database, we need a default server-side resulting table to save and present the extraction result after the partial information is extracted.

A resulting table comprises two parts: field name and field value. Field name is the defined name of target part and field value is the extracted value. In a resulting table, each column is a field and each row shows the extracted results of each target Web page, except that the first row displays the field names.

Fig. 8 is a resulting table of BBC News Click 2007 video archive [15]. The first column Date shows the date of target video. The second column Summary shows the summary of video and the third column Video shows the corresponding video address.

![Figure 8: A resulting table of BBC News Click 2007 video archive](image)

Through the interface, the Web service gets the request from the user. As the database receives SQL and searches for data in the tables, we parse the request to get the parameters and search for the desired information in the resulting table according to the parameters, then return the corresponding values to user in XML format. If the structure of desired field is continuous occurrence, the corresponding response data is the extracted value list.

5 Implementation and Evaluation

In this section, we implement our approach to construct the Web services.
For the dynamic URL type, we construct a Web service for CNN news search based on dynamic extraction. We search for the news in CNN and would get the result Web pages with similar layout no matter what keywords we enter. We can reuse the designed extraction pattern of a typical Web page to extract the partial information from these similar response Web pages. We developed a Web application for creating extraction pattern quickly and easily by GUI. We select the target part and the data type by mouse instead of parsing the HTML document of Web page to find out the paths manually.

1. We get a typical response Web page, and define the names and data types for each required part: news title part is \textit{NewsTitle} of continuous text type, news link part is \textit{NewsLink} of continuous link type, and publication date part is \textit{PublicationDate} of continuous text type. We use the following format to save the names, data types and paths of the selected parts as the extraction pattern.

\begin{verbatim}
<NewsTitle type="continuous text">BODY:0/DIV:1/DIV:0/DIV:2/DIV:0/A:0/></NewsTitle>
<NewsLink type="continuous link">BODY:0/DIV:1/DIV:0/DIV:2/DIV:0/A:0/></NewsLink>
<PublicationDate type="continuous text">BODY:0/DIV:1/DIV:0/DIV:2/DIV:0/SPAN:1/></PublicationDate>
\end{verbatim}

2. We configure the Web service interface, and give the parameter \textit{query}.

3. The Web service receives the request URL from the user and stimulate the submit to receive the response Web page.

\begin{verbatim}
http://VirtualWebServiceProxyServer/CNN/search?query=Tokyo
\end{verbatim}
Figure 10: A response of CNN news search from the Web service

4. The Web service extracts the partial information from the response Web page and returns the user the news about **Tokyo** in the first result page as shown in Fig. 10.

We can use the similar method to construct a Web service for ACM Calendar of Events [11]. The users can search for the events by date or text keyword as shown in Fig. 11.

Figure 11: ACM Calendar of Events and search result from constructed Web service

For the static URL type, we construct a Web service for querying information from BBC Country Profiles based on the static extraction. Compared with the dynamic extraction, we collect the URLs of target Web pages from the drop-down list and extract the selected parts irregularly to update the information about the new selected country leaders. Fig. 12 shows the integration of constructed Web services with Google Maps.

Our approach is applicable to the general Web applications and provides an extraction pattern creation tool with GUI. The extraction range includes text, link, picture and other
multimedia files of Web pages with static URLs or dynamic URLs, and our algorithm is applicable to the extraction from lists or tables in Web pages. We set the extraction frequencies for Web services and decrease the load of our proxy server. Although we realize the quick construction of Web services, our approach still depends on some manual work and is not robust enough when the Web applications change the layout of Web pages. During the interface configuration, it still needs some manual operations. Besides, the Web applications are designed for browsing by users, not for the parsing by computer program. It is difficult for us to select the desired parts from the Web pages of some Web sites because the desired parts and undesired parts are intermingled with each other and they have the similar paths.

6 Related Work

Taking advantage of the fact that a great and increasing number of Web sites have their pages automatically generated from database, a number of approaches have been proposed to analyze the structure of the Web pages of these Web sites with the purpose of manual or semi-automatic example-based information extraction and Web service construction ultimately.

To the information extraction, XSLT [6] uses the defined path patterns to find nodes that match with given paths repeatedly and outputs data by using information of the nodes and values of variables. Similarly, ANDES [9] is a XML-based methodology to use the manually created XSLT processors to realize the data extraction. PSO [14] is an approach to extract the parts of Web pages. It keeps the view information of the extracted parts by using the designated paths of tree structures of HTML documents. These Web knowledge extraction systems need the users to find out the paths of the desired parts from the HTML document by hands. IEPAD [1] proposes an automatic pattern discovery system. The users select the target pattern that contains desired information from the discovered candidate patterns, then the extractor receives the target pattern and a Web page as input, and applies pattern-matching algorithm to recognize and extract the information. [4] proposes an approach that provides the users a GUI to select the desired parts: text or image, uses the designated paths to extract the partial information from the similar Web pages, and finally returns the users a XML format result. However, this system does not support the link information extraction, which is important for the extraction of the information like the news search result of CNN.
To the Web service construction, HTML2RSS [10] is a Web service to automatically generate RSS feeds from HTML documents that consist of time-series items such as blog, BBS, chats and mailing lists. However, it is limited to the Web pages that consist of list of data items with similar data structures or special data structures. Toshiba Web Service Gateway [5] parses the response HTML page by flexible tree-style and returns the extracted data. It allows the users to try all the HTML parsers available one by one in the Web Service gateway to find which is the most suitable for parsing the response HTML pages. However, it is difficult for the users to develop the personalized parsers because the users need to find out the tags representing the desired parts one by one. GridXSLT [7] is an implementation of the XSLT language designed for large-scale parallel processing in grid computing environments. It provides a simplified approach to Web service development by implementation of XSLT documents, which need the programming of users. Pollock [8] can create a virtual Web service from FORM-based query interface of Web sites. It generates wrappers using XWrap, and WSDL file using Web site-related information, then publishes the details of the virtual Web service into UDDI, but this system needs the users to parse the HTML documents of the FORM-based Web pages.

Compared to these developed work, our approach realizes the extraction of almost all kinds of partial information such as text, object and link, and provides a GUI for easy part selection and data type definition. We can construct a Web service quickly and easily, and all the process of Web service construction does not need too much manual work and programming.

7 Conclusion and Future Work

Our approach is based on the partial information extraction, resulting table query and interface configuration. We transform the target Web page into the XML-format resulting table and respond to the requests of users with the field values. Our approach is applicable to the general Web applications including the information of link and multimedia files in table structure.

As future work, we will modify our approach to extend the types of applicable Web applications and the processes of similar paths reuse to intensify the extraction when Web sites change the layout of Web pages.

Finally, we will construct many types of Web services to develop the information integration systems by combination of the Web services.

References


